D 16

```
FFFFFFFFFFFFFFF        111              111           AAAAAAAAA
FFFFFFFFFFFFFFF        111              111           AAAAAAAAA
FFFFFFFFFFFFFFF        111              111           AAAAAAAAA
FFF                 111111           111111        AAA         AAA
FFF                 111111           111111        AAA         AAA
FFF                 111111           111111        AAA         AAA
FFF                    111              111        AAA         AAA
FFF                    111              111        AAA         AAA
FFF                    111              111        AAA         AAA
FFFFFFFFFFFF           111              111        AAA         AAA
FFFFFFFFFFFF           111              111        AAA         AAA
FFFFFFFFFFFF           111              111        AAA         AAA
FFF                    111              111        AAAAAAAAAAAAAAA
FFF                    111              111        AAAAAAAAAAAAAAA
FFF                    111              111        AAAAAAAAAAAAAAA
FFF                    111              111        AAA         AAA
FFF                    111              111        AAA         AAA
FFF                    111              111        AAA         AAA
FFF                 111111111        111111111     AAA         AAA
FFF                 111111111        111111111     AAA         AAA
FFF                 111111111        111111111     AAA         AAA
```

J 7

```
**FILE**ID**RDBLOK
```

```
RRRRRRRR    DDDDDDDD    BBBBBBBB    LL            OOOOOO    KK      KK
RRRRRRRR    DDDDDDDD    BBBBBBBB    LL            OOOOOO    KK      KK
RR      RR  DD      DD  BB      BB  LL          OO      OO  KK      KK
RR      RR  DD      DD  BB      BB  LL          OO      OO  KK      KK
RR      RR  DD      DD  BB      BB  LL          OO      OO  KK    KK
RR      RR  DD      DD  BB      BB  LL          OO      OO  KK    KK
RRRRRRRR    DD      DD  BBBBBBBB    LL          OO      OO  KKKKKK
RRRRRRRR    DD      DD  BBBBBBBB    LL          OO      OO  KKKKKK
RR  RR      DD      DD  BB      BB  LL          OO      OO  KK    KK
RR  RR      DD      DD  BB      BB  LL          OO      OO  KK    KK
RR    RR    DD      DD  BB      BB  LL          OO      OO  KK      KK       ....
RR    RR    DD      DD  BB      BB  LL          OO      OO  KK      KK       ....
RR      RR  DDDDDDDD    BBBBBBBB    LLLLLLLLLL    OOOOOO    KK      KK       ....
RR      RR  DDDDDDDD    BBBBBBBB    LLLLLLLLLL    OOOOOO    KK      KK       ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

RDB
V04

```
0 MODULE RDBLOK (
0                 LANGUAGE (BLISS32),
0                 IDENT = 'V04-000'
0                 ) =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:  F11ACP Structure Level 2
1 !
1 ! ABSTRACT:
1 !
1 !       This module contains routines for basic block I/O, as well
1 !       as the buffer management mechanism.
1 !
1 ! ENVIRONMENT:
1 !
1 !       STARLET operating system, including privileged system services
1 !       and internal exec routines.
1 !
1 !--
1
1 !
1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  13-Dec-1976  22:48
1 !
1 ! MODIFIED BY:
1 !
1 !       V02-003 ACG0157         Andrew C. Goldstein,    13-Mar-1980  14:43
1 !               Reverse LRU ordering of buffers in multi-block read
1 !
1 !       A0102   ACG0117         Andrew C. Goldstein,    16-Jan-1980  17:00
1 !               Return true I/O status on ACP I/O errors
1 !
```

```
;      58          0058  1 !          A0101   ACG0106          Andrew C. Goldstein,    15-Jan-1980  15:55
;      59          0059  1 !                  Change cache descriptor sizes to words
;      60          0060  1 !
;      61          0061  1 !          A0100   ACG00001         Andrew C. Goldstein,  10-Oct-1978  20:03
;      62          0062  1 !          Previous revision history moved to F11A.REV
;      63          0063  1 !**
;      64          0064  1
;      65          0065  1
;      66          0066  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;      67          0067  1 REQUIRE 'SRC$:FCPDEF.B32';
;      68          0382  1
;      69          0383  1
;      70          0384  1 FORWARD ROUTINE
;      71          0385  1         INIT_POOL         : NOVALUE,        ! initialize the buffer pool
;      72          0386  1         FIND_BUFFER,                        ! find an appropriate I/O buffer
;      73          0387  1         READ_BLOCK,                         ! read a block
;      74          0388  1         RESET_LBN         : NOVALUE,        ! assign new LBN to a buffer
;      75          0389  1         WRITE_BLOCK       : NOVALUE,        ! write a block
;      76          0390  1         CREATE_BLOCK,                       ! fabricate a buffer
;      77          0391  1         MARK_DIRTY        : NOVALUE,        ! mark buffer for write-back
;      78          0392  1         INVALIDATE        : NOVALUE,        ! invalidate a buffer
;      79          0393  1         WRITE_HEADER      : NOVALUE,        ! write file header
;      80          0394  1         FLUSH_BUFFERS     : NOVALUE,        ! flush all dirty buffers
;      81          0395  1         FLUSH_FID         : NOVALUE;        ! flush a file from the pool
```

```
!++
!
! Buffer pool data base.
!
! The root of the buffer data base is the pool vector which is used to index
! a block type into the buffer pool used for that type. The buffer pools are
! managed by 3 vectors, indexed by the pool code. The first vector contains
! the buffer index of the first buffer assigned to each pool. The second
! vector contains the number of buffers in each pool. The third vector
! contains the listheads for the LRU list of each pool.
!
! The buffers themselves are a block vector. Each buffer is identified by
! its address to the outside world, and internally by its vector index
! (the two are interchangeable in the obvious manner.) Associated with the
! buffers are status vectors: the UCB address of the currently resident
! block (0 if none), the LBN of the currently resident block, the LRU list
! entry, the file ID to which the block belongs, and the dirty bit.
!
!--

! Define the layout of the buffer pool. The pool descriptors are filled in
! by the pool initialization code. Note that each pool must consist of one
! virtually contiguous area. Note also that the storage map buffers are
! allocated first. This causes the buffer sweep at the end of each operation
! to write out the storage map blocks first, resulting in maximum safety.
!

LITERAL
        POOL_COUNT      = 3;                    ! number of pools

MACRO
        LRU_FLINK       = 0,0,32,0%,            ! LRU entry forward link
        LRU_BLINK       = 4,0,32,0%;            ! LRU entry back link

! Buffer pool vector
!

BIND
        POOL_TABLE      = UPLIT BYTE (  1,              ! file headers
                                        0,              ! storage map
                                        2,              ! directories
                                        1,              ! index file blocks
                                        2               ! random data blocks
                        ) : VECTOR [,BYTE];

! Base index of each buffer pool
!

OWN
        POOL_BASE       : VECTOR [POOL_COUNT, WORD];

! Number of buffers in each pool
!

OWN
        POOL_SIZE       : VECTOR [POOL_COUNT, WORD];
```

```
;  140          0453   1
;  141          0454   1  ! LRU list head for each pool
;  142          0455   1  !
;  143          0456   1
;  144          0457   1  OWN
;  145          0458   1          POOL_LRU        : BLOCKVECTOR [POOL_COUNT, 8, BYTE];
;  146          0459   1
;  147          0460   1  ! Pointers to buffer descriptor vectors. The vectors are dynamically allocated
;  148          0461   1  ! at initialization time.
;  149          0462   1  !
;  150          0463   1
;  151          0464   1  OWN
;  152          0465   1          BUFFER_LRU      : REF BLOCKVECTOR [, 8, BYTE],
;  153          0466   1          BUFFER_FID      : REF VECTOR,
;  154          0467   1          BUFFER_LBN      : REF VECTOR,
;  155          0468   1          BUFFER_UCB      : REF VECTOR,
;  156          0469   1          BUFFER_DIRTY    : REF BITVECTOR;
;  157          0470   1
;  158          0471   1  ! Pointer to the I/O buffers.
;  159          0472   1  !
;  160          0473   1
;  161          0474   1  STRUCTURE
;  162          0475   1          BUFVECTOR [I; N] =
;  163          0476   1          [N*512]
;  164          0477   1          (BUFVECTOR + I*512)<0, 32>;
;  165          0478   1
;  166          0479   1  OWN
;  167          0480   1          BUFFERS         : REF BUFVECTOR,
;  168          0481   1          BUFFER_COUNT;
;  170        0482  1 GLOBAL ROUTINE INIT_POOL : NOVALUE =
;  171        0483  1
;  172        0484  1 !++
;  173        0485  1 !
;  174        0486  1 ! FUNCTIONAL DESCRIPTION:
;  175        0487  1 !
;  176        0488  1 !       This routine initializes the buffer pool. It creates sufficient
;  177        0489  1 !       virtual space for the desired size buffer pool and sets up the
;  178        0490  1 !       descriptors.
;  179        0491  1 !
;  180        0492  1 ! CALLING SEQUENCE:
;  181        0493  1 !       INIT_POOL ()
;  182        0494  1 !
;  183        0495  1 ! INPUT PARAMETERS:
;  184        0496  1 !       NONE
;  185        0497  1 !
;  186        0498  1 ! IMPLICIT INPUTS:
;  187        0499  1 !       pool descriptor vectors
;  188        0500  1 !       ACP$GW_MAPCACHE: number of bitmap buffers to allocate
;  189        0501  1 !       ACP$GW_HDRCACHE: number of header buffers to allocate
;  190        0502  1 !       ACP$GW_DIRCACHE: number of directory buffers to allocate
;  191        0503  1 !
;  192        0504  1 ! OUTPUT PARAMETERS:
;  193        0505  1 !       NONE
;  194        0506  1 !
;  195        0507  1 ! IMPLICIT OUTPUTS:
;  196        0508  1 !       NONE
;  197        0509  1 !
;  198        0510  1 ! ROUTINE VALUE:
;  199        0511  1 !       NONE
;  200        0512  1 !
;  201        0513  1 ! SIDE EFFECTS:
;  202        0514  1 !       pool initialized
;  203        0515  1 !
;  204        0516  1 !--
;  205        0517  1
;  206        0518  2 BEGIN
;  207        0519  2
;  208        0520  2 LITERAL
;  209        0521  2         EXEC_MODE       = 1;            ! code for EXEC access mode
;  210        0522  2
;  211        0523  2 LOCAL
;  212        0524  2         MAP_COUNT,                      ! number of map buffers
;  213        0525  2         HDR_COUNT,                      ! number of header buffers
;  214        0526  2         DIR_COUNT,                      ! number of directory buffers
;  215        0527  2         BUFFER_SIZE,                    ! number of buffers in pool
;  216        0528  2         SIZE_NEEDED,                    ! total virtual space needed
;  217        0529  2         PAGE_COUNT,                     ! space actually obtained
;  218        0530  2         SPACE_DESC      : VECTOR [2];   ! descriptor of retun from $EXPREG
;  219        0531  2
;  220        0532  2 EXTERNAL
;  221        0533  2         ACP$GW_MAPCACHE : WORD ADDRESSING_MODE (ABSOLUTE),
;  222        0534  2                                         ! number of map buffers to use
;  223        0535  2         ACP$GW_HDRCACHE : WORD ADDRESSING_MODE (ABSOLUTE),
;  224        0536  2                                         ! number of header buffers to use
;  225        0537  2         ACP$GW_DIRCACHE : WORD ADDRESSING_MODE (ABSOLUTE);
;  226        0538  2                                         ! number of directory buffers to use
```

```
2
2 ! Compute the total virtual space needed and create it. The space needed is
2 ! the total number of buffers plus the descriptor space - 161 bits per buffer.
2 !
2
2 MAP_COUNT = MAXU (1, .ACP$GW_MAPCACHE);
2 HDR_COUNT = MAXU (1, .ACP$GW_HDRCACHE);
2 DIR_COUNT = MAXU (2, .ACP$GW_DIRCACHE);
2 BUFFER_SIZE = .MAP_COUNT + .HDR_COUNT + .DIR_COUNT;
2 SIZE_NEEDED = .BUFFER_SIZE + (.BUFFER_SIZE*161 + 4095) / 4096;
2
2 $EXPREG (PAGCNT = .SIZE_NEEDED, ACMODE = EXEC_MODE, RETADR = SPACE_DESC);
2
2 ! Compute the space we actually got and make sure it is at least the minimum.
2 ! If it is less then we asked for, divide it up in the ratio 1:1:6.
2 !
2
2 PAGE_COUNT = (.SPACE_DESC[1] - .SPACE_DESC[0]) / 512 + 1;
2 IF .PAGE_COUNT LSSU 5
2 THEN $EXIT (CODE = SS$_INSFMEM);
2
2 IF .PAGE_COUNT LSSU .SIZE_NEEDED
2 THEN
3     BEGIN
3     BUFFER_SIZE = (.PAGE_COUNT*4096) / 4257;
3     MAP_COUNT = HDR_COUNT = .BUFFER_SIZE / 8;
3     DIR_COUNT = .BUFFER_SIZE - (.MAP_COUNT + .HDR_COUNT);
2     END;
2
2 ! Allocate and set up the pointers for the buffer descriptors and the buffers
2 ! themselves.
2 !
2
2 BUFFER_LRU = .SPACE_DESC[0];
2 BUFFER_FID = .BUFFER_LRU + .BUFFER_SIZE*8;
2 BUFFER_LBN = .BUFFER_FID + .BUFFER_SIZE*4;
2 BUFFER_UCB = .BUFFER_LBN + .BUFFER_SIZE*4;
2 BUFFER_DIRTY = .BUFFER_UCB + .BUFFER_SIZE*4;
2 BUFFERS = .BUFFER_DIRTY + (.BUFFER_SIZE+7)/8 + 511 AND NOT 511;
2
2 POOL_SIZE[0] = .MAP_COUNT;
2 POOL_SIZE[1] = .HDR_COUNT;
2 POOL_SIZE[2] = .DIR_COUNT;
2 POOL_BASE[0] = 0;
2 POOL_BASE[1] = .MAP_COUNT;
2 POOL_BASE[2] = .MAP_COUNT + .HDR_COUNT;
2 BUFFER_COUNT = .BUFFER_SIZE;
2
2 ! Loop for all pools. First init the LRU list head to be empty. Then loop
2 ! for all buffers in each pool, linking each buffer into the pool LRU listhead.
2 !
2
2 INCR POOL FROM 0 TO POOL_COUNT-1 DO
3     BEGIN
3     POOL_LRU[.POOL, LRU_FLINK] = POOL_LRU[.POOL, LRU_FLINK];
3     POOL_LRU[.POOL, LRU_BLINK] = POOL_LRU[.POOL, LRU_FLINK];
3
```

```
;    284          0596  3        INCR I FROM 0 TO .POOL_SIZE[.POOL]-1 DO
;    285          0597  3            INSQUE (BUFFER_LRU[.POOL_BASE[.POOL]+.I, LRU_FLINK],
;    286          0598  3                    .POOL_[RU[.POOL, LRU_BLINK]);
;    287          0599  2        END;
;    288          0600  2
;    289          0601  1 END;                                        ! end of routine INIT_POOL


                                                                  .TITLE  RDBLOK
                                                                  .IDENT  \V04-000\

                                                                  .PSECT  $CODE$,NOWRT,2

                          02  01  02  00  01  00000 P.AAA:   .BYTE   1, 0, 2, 1, 2                                   ;

                                                                  .PSECT  $LOCKEDD1$,NOEXE,2

                                              00000 POOL_BASE:
                                                                  .BLKB   6
                                              00006               .BLKB   2
                                              00008 POOL_SIZE:
                                                                  .BLKB   6
                                              0000E               .BLKB   2
                                              00010 POOL_LRU:
                                                                  .BLKB   24
                                              00028 BUFFER_LRU:
                                                                  .BLKB   4
                                              0002C BUFFER_FID:
                                                                  .BLKB   4
                                              00030 BUFFER_LBN:
                                                                  .BLKB   4
                                              00034 BUFFER_UCB:
                                                                  .BLKB   4
                                              00038 BUFFER_DIRTY:
                                                                  .BLKB   4
                                              0003C BUFFERS:.BLKB   4
                                              00040 BUFFER_COUNT:
                                                                  .BLKB   4

                                                    POOL_TABLE=           P.AAA
                                                                  .EXTRN  ACP$GW_MAPCACHE
                                                                  .EXTRN  ACP$GW_HDRCACHE
                                                                  .EXTRN  ACP$GW_DIRCACHE
                                                                  .EXTRN  SYS$EXPREG, SYS$EXIT

                                                                  .PSECT  $CODE$,NOWRT,2

                                        01FC  00000               .ENTRY  INIT_POOL, Save R2,R3,R4,R5,R6,R7,R8          ; 0482
                          58      0000'   CF   9E  00002               MOVAB   BUFFER_LRU, R8                                 ;
                          5E              08   C2  00007               SUBL2   #8, SP                                         ;
                          50  00000000G   9F   3C  0000A               MOVZWL  @#ACP$GW_MAPCACHE, R0                         ; 0544
                                          03   12  00011               BNEQ    1$                                             ;
                          50              01   D0  00013               MOVL    #1, R0                                         ;
                          54              50   D0  00016 1$:           MOVL    R0, MAP_COUNT                                  ;
                          50  00000000G   9F   3C  00019               MOVZWL  @#ACP$G@_HDRCACHE, R0                         ; 0545
                                          03   12  00020               BNEQ    2$                                             ;
                          50              01   D0  00022               MOVL    #1, R0                                         ;
```

```
                  56             50  D0 00025 2$:    MOVL    R0, HDR_COUNT                                    ;
                  50 00000000G   9F  3C 00028        MOVZWL  @#ACP$GW_DIRCACHE, R0                            ; 0546
                  02             50  B1 0002F        CMPW    R0, #2                                           ;
                                 03  1E 00032        BGEQU   3$                                               ;
                  50             02  D0 00034        MOVL    #2, R0                                           ;
                  57             50  D0 00037 3$:    MOVL    R0, DIR_COUNT                                    ;
    50            54             56  C1 0003A        ADDL3   HDR_COUNT, MAP_COUNT, R0                         ; 0547
    52            50             57  C1 0003E        ADDL3   DIR_COUNT, R0, BUFFER_SIZE                       ;
    50            52 000000A1    8F  C5 00042        MULL3   #161, BUFFER_SIZE, R0                            ; 0548
                  50     0FFF    C0  9E 0004A        MOVAB   4095(R0), R0                                     ;
                  50 00001000    8F  C6 0004F        DIVL2   #4096, R0                                        ;
    55            50             52  C1 00056        ADDL3   BUFFER_SIZE, R0, SIZE_NEEDED                     ;
                  7E             01  7D 0005A        MOVQ    #1, -(SP)                                        ; 0550
                           08    AE  9F 0005D        PUSHAB  SPACE_DESC                                       ;
                                 55  DD 00060        PUSHL   SIZE_NEEDED                                      ;
       00000000G  00             04  FB 00062        CALLS   #4, SYS$EXPREG                                   ;
    53       04   AE             6E  C3 00069        SUBL3   SPACE_DESC, SPACE_DESC+4, R3                     ; 0556
                  53 00000200    8F  C6 0006E        DIVL2   #512, R3                                         ;
                                 53  D6 00075        INCL    PAGE_COUNT                                       ;
                  05             53  D1 00077        CMPL    PAGE_COUNT, #5                                   ; 0557
                                 0C  1E 0007A        BGEQU   4$                                               ;
                  7E     0124    8F  3C 0007C        MOVZWL  #292, -(SP)                                      ; 0558
       00000000G  00             01  FB 00081        CALLS   #1, SYS$EXIT                                     ;
                  55             53  D1 00088 4$:    CMPL    PAGE_COUNT, SIZE_NEEDED                          ; 0560
                                 1B  1E 0008B        BGEQU   5$                                               ;
    53            53             0C  78 0008D        ASHL    #12, R3, R3                                      ; 0563
    52            53 000010A1    8F  C7 00091        DIVL3   #4257, R3, BUFFER_SIZE                           ;
    56            52             08  C7 00099        DIVL3   #8, BUFFER_SIZE, HDR_COUNT                       ; 0564
                  54             56  D0 0009D        MOVL    HDR_COUNT, MAP_COUNT                             ;
    50            54             56  C1 000A0        ADDL3   HDR_COUNT, MAP_COUNT, R0                         ; 0565
    57            52             50  C3 000A4        SUBL3   R0, BUFFER_SIZE, DIR_COUNT                       ;
                  68             6E  D0 000A8 5$:    MOVL    SPACE_DESC, BUFFER_LRU                           ; 0572
             04   A8       00  B842  7E 000AB        MOVAQ   @BUFFER_LRU[BUFFER_SIZE], BUFFER_FID             ; 0573
             08   A8       04  B842  DE 000B1        MOVAL   @BUFFER_FID[BUFFER_SIZE], BUFFER_LBN             ; 0574
             0C   A8       08  B842  DE 000B7        MOVAL   @BUFFER_LBN[BUFFER_SIZE], BUFFER_UCB             ; 0575
             10   A8       0C  B842  DE 000BD        MOVAL   @BUFFER_UCB[BUFFER_SIZE], BUFFER_DIRTY           ; 0576
                  50       07    A2  9E 000C3        MOVAB   7(R2), R0                                        ; 0577
                  50             08  C6 000C7        DIVL2   #8, R0                                           ;
                  50       10    A8  C0 000CA        ADDL2   BUFFER_DIRTY, R0                                 ;
                  50     01FF    C0  9E 000CE        MOVAB   511(R0), R0                                      ;
14  A8            50 000001FF    8F  CB 000D3        BICL3   #511, R0, BUFFERS                                ;
             E0   A8             54  B0 000DC        MOVW    MAP_COUNT, POOL_SIZE                             ; 0579
             E2   A8             56  B0 000E0        MOVW    HDR_COUNT, POOL_SIZE+2                           ; 0580
             E4   A8             57  B0 000E4        MOVW    DIR_COUNT, POOL_SIZE+4                           ; 0581
                           D8    A8  B4 000E8        CLRW    POOL_BASE                                        ; 0582
             DA   A8             54  B0 000EB        MOVW    MAP_COUNT, POOL_BASE+2                           ; 0583
DC  A8            54             56  A1 000EF        ADDW3   HDR_COUNT, MAP_COUNT, POOL_BASE+4                ; 0584
             18   A8             52  D0 000F4        MOVL    BUFFER_SIZE, BUFFER_COUNT                        ; 0585
                                 50  D4 000F8        CLRL    POOL                                             ; 0591
                  51       E8  A840  7E 000FA 6$:    MOVAQ   POOL_LRU[POOL], R1                               ; 0593
                  61             51  D0 000FF        MOVL    R1, (R1)                                         ;
                           EC  A840  7F 00102        PUSHAQ  POOL_LRU+4[POOL]                                 ; 0594
                  9E             51  D0 00106        MOVL    R1, @(SP)+                                       ;
                  53       E0  A840  3C 00109        MOVZWL  POOL_SIZE[POOL], R3                              ; 0596
                  52             01  CE 0010E        MNEGL   #1, I                                            ;
                                 15  11 00111        BRB     8$                                               ;
                  51       D8  A840  3C 00113 7$:    MOVZWL  POOL_BASE[POOL], R1                              ; 0597
```

```
                         51        52  C0 00118          ADDL2   I, R1                                            ;
                         54  EC A840  7E 0011B          MOVAQ   POOL_LRU+4[POOL], R4                             ; 0598
                             00 B841  7F 00120          PUSHAQ  @BUFFER_LRU[R1]                                  ;
                   00    B4        9E  0E 00124          INSQUE  @(SP)+, @0(R4)                                   ;
              E7         52        53  F2 00128  8$:     AOBLSS  R3, I, 7$                                        ; 0597
              CA         50        02  F3 0012C          AOBLEQ  #2, POOL, 6$                                     ; 0591
                                       04 00130          RET                                                      ; 0601

; Routine Size:  305 bytes,    Routine Base:  $CODE$ + 0005
```

```
1 ROUTINE FIND_BUFFER (LBN, TYPE, COUNT, FOUND_COUNT) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine searches for a buffer suitable for the indicated
1 !       block(s). It looks first for a buffer containing that block; failing
1 !       that, it finds free buffers or frees them.
1 !
1 ! CALLING SEQUENCE:
1 !       FIND_BUFFER (ARG1, ARG2, ARG3, ARG4)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: LBN of first desired block
1 !       ARG2: type code of buffer
1 !       ARG3: length of buffer desired in blocks
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_UCB: UCB of device in use
1 !       DIR_FCB: FCB of directory file
1 !
1 ! OUTPUT PARAMETERS:
1 !       ARG4: number of blocks of buffer reserved
1 !
1 ! IMPLICIT OUTPUTS:
1 !       BUFFER_LBN: (of returned buffer(s)) LBN of block
1 !       BUFFER_UCB: (of returned buffer(s)) CURRENT_UCB if block was resident,
1 !               zero if new buffer
1 !
1 ! ROUTINE VALUE:
1 !       index of first buffer found
1 !
1 ! SIDE EFFECTS:
1 !       LRU list relinked, buffers may be written
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       I,                                      ! index of found buffer
2       N,                                      ! number of found buffers
2       POOL,                                   ! index of pool to use
2       NEXT_LBN,                               ! next higher LBN in pool
2       LRU_ENTRY       : REF BLOCK;            ! pointer to buffer LRU entry
2
2 EXTERNAL
2       CURRENT_UCB     : REF BBLOCK,           ! UCB of current device
2       CURRENT_VCB     : REF BBLOCK,           ! VCB of current device
2       CURRENT_FIB     : REF BBLOCK,           ! address of FIB of current operation
2       PMS_TOT_CACHE,                          ! cumulative count of buffer cache hits
2       DIR_FCB         : REF BBLOCK,           ! directory FCB
2       ACP$GB_MAXREAD  : BYTE ADDRESSING_MODE (ABSOLUTE);
2                                               ! maximum number of blocks to read
2
2
```

```
2 ! First search the indicated buffer pool for a buffer containing the
2 ! desired LBN and UCB. Also track the LBN of the next highest block in the
2 ! cache. Note that we assume that block type classes are
2 ! nonintersecting sets, and thus avoid having the same block show up in
2 ! multiple pools by good behavior in the file system.
2 !
2
2 POOL = .POOL_TABLE[.TYPE];
2 NEXT_LBN = -1;
2
3 I = (
3     INCR J FROM .POOL_BASE[.POOL] TO .POOL_BASE[.POOL] + .POOL_SIZE[.POOL] - 1
3     DO
3     IF .BUFFER_UCB[.J] EQL .CURRENT_UCB
3     THEN
4         BEGIN
4         IF  .BUFFER_LBN[.J] GEQU .LBN
4         AND .BUFFER_LBN[.J] LSSU .NEXT_LBN
4         THEN NEXT_LBN = .BUFFER_LBN[.J];
4
4         IF .BUFFER_LBN[.J] EQL .LBN
4         THEN EXITLOOP .J
4         END
2     );
2
2 ! If we found a block, pull the buffer out of the LRU and count a cache hit.
2 ! Link the buffer onto the end of the LRU list to indicate recent use.
2 ! On a cache hit, we always return exactly one block.
2 !
2
2 IF .I NEQ -1
2 THEN
3     BEGIN
3     REMQUE (BUFFER_LRU[.I, LRU_FLINK], LRU_ENTRY);
3     INSQUE (.LRU_ENTRY, .POOL_LRU[.POOL, LRU_BLINK]);
3     PMS_TOT_CACHE = .PMS_TOT_CACHE + 1;
3     .FOUND_COUNT = 1;
3     END
3
3 ! Get the first buffer on the LRU. If multiple buffers are requested,
3 ! grab additional buffers in ascending memory order until we hit the end of the
3 ! pool. Stop if we hit a block that is already in the cache (recorded by
3 ! NEXT_LBN). If we still need more, get them in descending memory order. Then
3 ! loop for all found buffers, relinking them onto the LRU in ascending
3 ! order and writing them if they are dirty.
3 !
3
2 ELSE
3     BEGIN
3     I = (.POOL_LRU[.POOL, LRU_FLINK] - BUFFER_LRU[0, LRU_FLINK]) / 8;
3
3     N = .COUNT;
3     IF .N GTRU .ACP$GB_MAXREAD
3     THEN N = .ACP$GB_MAXREAD;
3     IF .NEXT_LBN - .LBN LEQU .N
3     THEN N = .NEXT_LBN - .LBN;
3
```

```
;  405        0716  3      IF .POOL_SIZE[.POOL] + .POOL_BASE[.POOL] - .I LSS .N
;  406        0717  3      THEN
;  407        0718  4          BEGIN
;  408        0719  4          IF .POOL_SIZE[.POOL] LEQ .N
;  409        0720  4          THEN
;  410        0721  5              BEGIN
;  411        0722  5              I = .POOL_BASE[.POOL];
;  412        0723  5              N = .POOL_SIZE[.POOL];
;  413        0724  5              END
;  414        0725  4          ELSE
;  415        0726  4              I = .POOL_SIZE[.POOL] + .POOL_BASE[.POOL] - .N;
;  416        0727  3          END;
;  417        0728  3      .FOUND_COUNT = .N;
;  418        0729  3
;  419        0730  3      DECR J FROM .N-1 TO 0
;  420        0731  3      DO
;  421        0732  4          BEGIN
;  422        0733  4          REMQUE (BUFFER_LRU[.I+.J, LRU_FLINK], LRU_ENTRY);
;  423        0734  4          INSQUE (.LRU_ENTRY, .POOL_LRU[.POOL, LRU_BLINK]);
;  424        0735  4
;  425        0736  4          IF .BUFFER_DIRTY[.I+.J]
;  426        0737  4          THEN WRITE_BLOCK (BUFFERS[.I+.J]);
;  427        0738  4
;  428        0739  4          BUFFER_UCB[.I+.J] = 0;
;  429        0740  4          BUFFER_LBN[.I+.J] = .LBN + .J;
;  430        0741  4
;  431        0742  4          CASE .TYPE FROM 0 TO 4 OF
;  432        0743  4          SET
;  433        0744  4          [INDEX_TYPE, HEADER_TYPE]:  BUFFER_FID[.I+.J] = 1;
;  434        0745  4          [BITMAP_TYPE]:              BUFFER_FID[.I+.J] = 2;
;  435        0746  5          [DIRECTORY_TYPE]:           BEGIN
;  436        0747  5                                      BUFFER_FID[.I+.J] = .DIR_FCB[FCB$W_FID_NUM];
;  437        0748  5                                      IF .CURRENT_VCB[VCB$V_EXTFID]
;  438        0749  5                                      THEN (BUFFER_FID[.I+.J])<16,8> = .DIR_FCB[FCB$B_FID_NMX];
;  439        0750  4                                      END;
;  440        0751  5          [DATA_TYPE]:                BEGIN
;  441        0752  5                                      BUFFER_FID[.I+.J] = .CURRENT_FIB[FIB$W_FID_NUM];
;  442        0753  5                                      IF .CURRENT_VCB[VCB$V_EXTFID]
;  443        0754  5                                      THEN (BUFFER_FID[.I+.J])<16,8> = .CURRENT_FIB[FIB$B_FID_NMX];
;  444        0755  4                                      END;
;  445        0756  4          [OUTRANGE]:      (BUG_CHECK (BADBUFTYP, FATAL, 'Bad ACP buffer type code'); 0);
;  446        0757  4          TES;
;  447        0758  3          END;
;  448        0759  2      END;
;  449        0760  2
;  450        0761  2  RETURN .I;
;  451        0762  2
;  452        0763  1  END;                                    ! end of routine FIND_BUFFER


                                                                         .EXTRN  CURRENT_UCB, CURRENT_VCB
                                                                         .EXTRN  CURRENT_FIB, PMS_TOT_CACHE
                                                                         .EXTRN  DIR_FCB, ACP$GB_MAXREAD
                                                                         .EXTRN  BUG$_BADBUFTYP

                                                         01FC 00000 FIND_BUFFER:
```

```
                                                     .WORD    Save R2,R3,R4,R5,R6,R7,R8              ; 0602
                  58      0000'   CF   9E 00002      MOVAB    BUFFER_FID, R8                         ;
                  50      FEBF    CF   9E 00007      MOVAB    POOL_TABLE, R0                         ; 0666
                  55        08 BC40    9A 0000C      MOVZBL   @TYPE[R0], POOL                        ;
                  51              01   CE 00011      MNEGL    #1, NEXT_LBN                           ; 0667
                  54        D4 A845    3C 00014      MOVZWL   POOL_BASE[POOL], R4                    ; 0670
                  53        DC A845    3C 00019      MOVZWL   POOL_SIZE[POOL], R3                    ;
       56         54              53   C1 0001E      ADDL3    R3, R4, R6                             ;
                  50        FF    A4   9E 00022      MOVAB    -1(R4), J                              ; 0672
                                  27   11 00026      BRB      3$                                     ;
           0000G  CF        08 B840    D1 00028 1$:  CMPL     @BUFFER_UCB[J], CURRENT_UCB            ;
                                  1E   12 0002F      BNEQ     3$                                     ;
                  52        04 B840    D0 00031      MOVL     @BUFFER_LBN[J], R2                     ; 0675
            04    AC              52   D1 00036      CMPL     R2, LBN                                ;
                                  08   1F 0003A      BLSSU    2$                                     ;
                  51              52   D1 0003C      CMPL     R2, NEXT_LBN                           ; 0676
                                  03   1E 0003F      BGEQU    2$                                     ;
                  51              52   D0 00041      MOVL     R2, NEXT_LBN                           ; 0677
            04    AC              52   D1 00044 2$:  CMPL     R2, LBN                                ; 0679
                                  05   12 00048      BNEQ     3$                                     ;
                  52              50   D0 0004A      MOVL     J, I                                   ; 0680
                                  07   11 0004D      BRB      4$                                     ;
       D5         50              56   F2 0004F 3$:  AOBLSS   R6, J, 1$                              ; 0672
                  52              01   CE 00053      MNEGL    #1, I                                  ; 0670
        FFFFFFFF  8F              52   D1 00056 4$:  CMPL     I, #-1                                 ; 0689
                                  1C   13 0005D      BEQL     5$                                     ;
                  50        FC B842    7E 0005F      MOVAQ    @BUFFER_LRU[I], R0                     ; 0692
                  57              60   0F 00064      REMQUE   (R0), LRU_ENTRY                        ;
                  50        E8 A845    7E 00067      MOVAQ    POOL_LRU+4[POOL], R0                   ; 0693
            00    B0              67   0E 0006C      INSQUE   (LRU_ENTRY), @0(R0)                    ;
                          0000G   CF   D6 00070      INCL     PMS_TOT_CACHE                          ; 0694
            10    BC              01   D0 00074      MOVL     #1, @FOUND_COUNT                       ; 0695
                                00F4   31 00078      BRW      20$                                    ; 0689
                            E4 A845    7F 0007B 5$:  PUSHAQ   POOL_LRU[POOL]                         ; 0708
       50         9E        FC    A8   C3 0007F      SUBL3    BUFFER_LRU, @(SP)+, R0                 ;
       52         50              08   C7 00084      DIVL3    #8, R0, I                              ;
                  50        0C    AC   D0 00088      MOVL     COUNT, N                               ; 0710
                  56 00000000G    9F   9A 0008C      MOVZBL   @#ACP$GB_MAXREAD, R6                   ; 0711
                  56              50   D1 00093      CMPL     N, R6                                  ;
                                  03   1B 00096      BLEQU    6$                                     ;
                  50              56   D0 00098      MOVL     R6, N                                  ; 0712
                  51        04    AC   C2 0009B 6$:  SUBL2    LBN, R1                                ; 0713
                  50              51   D1 0009F      CMPL     R1, N                                  ;
                                  03   1A 000A2      BGTRU    7$                                     ;
                  50              51   D0 000A4      MOVL     R1, N                                  ; 0714
       51         53              54   C1 000A7 7$:  ADDL3    R4, R3, R1                             ; 0716
       56         51              52   C3 000AB      SUBL3    I, R1, R6                              ;
                  50              56   D1 000AF      CMPL     R6, N                                  ;
                                  11   18 000B2      BGEQ     9$                                     ;
                  50              53   D1 000B4      CMPL     R3, N                                  ; 0719
                                  08   14 000B7      BGTR     8$                                     ;
                  52              54   D0 000B9      MOVL     R4, I                                  ; 0722
                  50              53   D0 000BC      MOVL     R3, N                                  ; 0723
                                  04   11 000BF      BRB      9$                                     ; 0719
       52         51              50   C3 000C1 8$:  SUBL3    N, R1, I                               ; 0726
            10    BC              50   D0 000C5 9$:  MOVL     N, @FOUND_COUNT                        ; 0728
                  53              50   D0 000C9      MOVL     N, J                                   ; 0730
```

```
                                                     78  11   000CC         BRB      16$                                          ;
                          50            52           53  C1   000CE 10$:    ADDL3    J, I, R0                                     ; 0733
                                        50      FC B840  7E   000D2         MOVAQ    @BUFFER_LRU[R0], R0                          ;
                                        57           60  0F   000D7         REMQUE   (R0), LRU_ENTRY                              ;
                                        50      E8 A845  7E   000DA         MOVAQ    POOL_LRU+4[POOL], R0                         ; 0734
                                   00   B0           67  0E   000DF         INSQUE   (LRU_ENTRY), @0(R0)                          ;
                          54            52           53  C1   000E3         ADDL3    J, I, R4                                     ; 0736
                          0D       0C   B8           54  E1   000E7         BBC      R4, @BUFFER_DIRTY, 11$                       ;
                          50            54           09  78   000EC         ASHL     #9, R4, R0                                   ; 0737
                                                10 B840  9F   000F0         PUSHAB   @BUFFERS[R0]                                 ;
                                0000V   CF           01  FB   000F4         CALLS    #1, WRITE_BLOCK                              ;
                                                08 B844  D4   000F9 11$:    CLRL     @BUFFER_UCB[R4]                              ; 0739
                                   04 B844      04 BC43  9E   000FD         MOVAB    @LBN[J], @BUFFER_LBN[R4]                     ; 0740
                          04            00      08   AC  CF   00104         CASEL    TYPE, #0, #4                                 ; 0742
         0010           001E          0017         0010       00109 12$:    .WORD    13$-12$,-                                    ;
                                                   003F       00111                  14$-12$,-                                    ;
                                                                                     15$-12$,-                                    ;
                                                                                     13$-12$,-                                    ;
                                                                                     17$-12$                                      ;
                                                       FEFF   00113         BUGW                                                  ; 0756
                                                      0000*   00115         .WORD    <BUG$_BADBUFTYP!4>                           ;
                                                     4E  11   00117         BRB      18$                                          ;
                                   00 B844           01  D0   00119 13$:    MOVL     #1, @BUFFER_FID[R4]                          ; 0744
                                                     47  11   0011E         BRB      18$                                          ;
                                   00 B844           02  D0   00120 14$:    MOVL     #2, @BUFFER_FID[R4]                          ; 0745
                                                     40  11   00125         BRB      18$                                          ;
                                        50   0000G   CF  D0   00127 15$:    MOVL     DIR_FCB, R0                                  ; 0747
                                   00 B844      24   A0  3C   0012C         MOVZWL   36(R0), @BUFFER_FID[R4]                      ;
                                        51   0000G   CF  D0   00132         MOVL     CURRENT_VCB, R1                              ; 0748
                          2B       0B   A1           05  E1   00137         BBC      #5, 11(R1), 18$                              ;
                                                00 B844  DF   0013C         PUSHAL   @BUFFER_FID[R4]                              ; 0749
           9E             08            10      29   A0  F0   00140         INSV     41(R0), #16, #8, @(SP)+                      ;
                                                     1F  11   00146 16$:    BRB      18$                                          ; 0742
                                        50   0000G   CF  D0   00148 17$:    MOVL     CURRENT_FIB, R0                              ; 0752
                                   00 B844      04   A0  3C   0014D         MOVZWL   4(R0), @BUFFER_FID[R4]                       ;
                                        51   0000G   CF  D0   00153         MOVL     CURRENT_VCB, R1                              ; 0753
                          0A       0B   A1           05  E1   00158         BBC      #5, 11(R1), 18$                              ;
                                                00 B844  DF   0015D         PUSHAL   @BUFFER_FID[R4]                              ; 0754
           9E             08            10      09   A0  F0   00161         INSV     9(R0), #16, #8, @(SP)+                       ;
                                        02           53  F4   00167 18$:    SOBGEQ   J, 19$                                       ; 0730
                                                     03  11   0016A         BRB      20$                                          ;
                                                   FF5F  31   0016C 19$:    BRW      10$                                          ;
                                        50           52  D0   0016F 20$:    MOVL     I, R0                                        ; 0761
                                                         04   00172         RET                                                   ; 0763

; Routine Size:  371 bytes,    Routine Base:  $CODE$ + 0136
;  454         0764  1 GLOBAL ROUTINE READ_BLOCK (LBN, COUNT, TYPE) =
;  455         0765  1
;  456         0766  1 !++
;  457         0767  1 !
;  458         0768  1 ! FUNCTIONAL DESCRIPTION:
;  459         0769  1 !
;  460         0770  1 !       This routine reads the desired block(s) from the disk.
;  461         0771  1 !       Blocks are categorized by type to aid buffer management.
;  462         0772  1 !       Note that the caller assumes only one block is ever read; multiple
;  463         0773  1 !       blocks read ahead are acquired through cache hits on subsequent calls.
;  464         0774  1 !
;  465         0775  1 ! CALLING SEQUENCE:
;  466         0776  1 !       READ_BLOCK (ARG1, ARG2, ARG3)
;  467         0777  1 !
;  468         0778  1 ! INPUT PARAMETERS:
;  469         0779  1 !       ARG1: LBN of block(s)
;  470         0780  1 !       ARG2: number of blocks to read
;  471         0781  1 !       ARG3: block type code
;  472         0782  1 !
;  473         0783  1 ! IMPLICIT INPUTS:
;  474         0784  1 !       CURRENT_UCB contains address of UCB in process
;  475         0785  1 !
;  476         0786  1 ! OUTPUT PARAMETERS:
;  477         0787  1 !       NONE
;  478         0788  1 !
;  479         0789  1 ! IMPLICIT OUTPUTS:
;  480         0790  1 !       IO_STATUS receives status of I/O transfer
;  481         0791  1 !
;  482         0792  1 ! ROUTINE VALUE:
;  483         0793  1 !       address of buffer containing block
;  484         0794  1 !
;  485         0795  1 ! SIDE EFFECTS:
;  486         0796  1 !       BLOCK READ
;  487         0797  1 !
;  488         0798  1 !--
;  489         0799  1
;  490         0800  2 BEGIN
;  491         0801  2
;  492         0802  2 LOCAL
;  493         0803  2         I,                                      ! index of buffer used
;  494         0804  2         STATUS,                                 ! QIO service status
;  495         0805  2         FOUND_COUNT;                            ! count of buffers gotten
;  496         0806  2
;  497         0807  2 EXTERNAL
;  498         0808  2         PMS_TOT_READ,                           ! cumulative count of disk reads
;  499         0809  2         CLEANUP_FLAGS   : BITVECTOR,            ! cleanup action flags
;  500         0810  2         DIR_VBN,                                ! current VBN in directory buffer
;  501         0811  2         BITMAP_VBN,                             ! current VBN in storage map buffer
;  502         0812  2         IO_CHANNEL,                             ! channel number for all I/O
;  503         0813  2         CURRENT_UCB,                            ! UCB of device in process
;  504         0814  2         IO_STATUS       : VECTOR;               ! common I/O status block
;  505         0815  2
;  506         0816  2
;  507         0817  2 ! Find a suitable block buffer. If it does not already contain the block,
;  508         0818  2 ! read it.
;  509         0819  2 !
;  510         0820  2
;  511          0821  2 I = FIND_BUFFER (.LBN, .TYPE, .COUNT, FOUND_COUNT);
;  512          0822  2
;  513          0823  2 IF .BUFFER_UCB[.I] EQL 0
;  514          0824  2 THEN
;  515          0825  3     BEGIN
;  516          0826  3     PMS_TOT_READ = .PMS_TOT_READ + 1;
;  517        P 0827  3     STATUS = $QIOW (
;  518        P 0828  3                 EFN   = EFN,
;  519        P 0829  3                 CHAN  = .IO_CHANNEL,
;  520        P 0830  3                 FUNC  = IO$_READLBLK,
;  521        P 0831  3                 IOSB  = IO_STATUS,
;  522        P 0832  3                 P1    = BUFFERS[.I],
;  523        P 0833  3                 P2    = .FOUND_COUNT*512,
;  524        P 0834  3                 P3    = .LBN
;  525          0835  3                 );
;  526          0836  3     IF NOT .STATUS THEN IO_STATUS = .STATUS;
;  527          0837  3     IF NOT .IO_STATUS
;  528          0838  3     THEN
;  529          0839  4         BEGIN
;  530          0840  4         INCR J FROM 0 TO .FOUND_COUNT-1
;  531          0841  4         DO
;  532          0842  4             INVALIDATE (BUFFERS[.I+.J]);
;  533          0843  4         DIR_VBN = 0;
;  534          0844  4         BITMAP_VBN = 0;
;  535          0845  4         ERR_EXIT (.IO_STATUS<0,16>);
;  536          0846  3         END;
;  537          0847  3     INCR J FROM 0 TO .FOUND_COUNT - 1
;  538          0848  3     DO
;  539          0849  3         BUFFER_UCB[.I+.J] = .CURRENT_UCB;
;  540          0850  2     END;
;  541          0851  2
;  542          0852  2 RETURN BUFFERS[.I];
;  543          0853  2
;  544          0854  1 END;                                        ! end of routine READ_BLOCK


                                                            .EXTRN  PMS_TOT_READ, CLEANUP_FLAGS
                                                            .EXTRN  DIR_VBN, BITMAP_VBN
                                                            .EXTRN  IO_CHANNEL, IO_STATUS
                                                            .EXTRN  SYS$QIOW

                                 003C 00000                 .ENTRY  READ_BLOCK, Save R2,R3,R4,R5          ; 0764
                   55    0000G  CF   9E 00002               MOVAB   IO_STATUS, R5                         ;
                   54    0000'  CF   9E 00007               MOVAB   BUFFERS, R4                           ;
                   5E            04  C2 0000C               SUBL2   #4, SP                                ;
                                 5E  DD 0000F               PUSHL   SP                                    ; 0821
                           08    AC  DD 00011               PUSHL   COUNT                                 ;
                           0C    AC  DD 00014               PUSHL   TYPE                                  ;
                           04    AC  DD 00017               PUSHL   LBN                                   ;
              FE6E   CF          04  FB 0001A               CALLS   #4, FIND_BUFFER                       ;
                     53          50  D0 0001F               MOVL    R0, I                                 ;
                           F8  B443  D5 00022               TSTL    @BUFFER_UCB[I]                        ; 0823
                                 6D  12 00026               BNEQ    7$                                    ;
                         0000G   CF  D6 00028               INCL    PMS_TOT_READ                          ; 0826
                                 7E  7C 0002C               CLRQ    -(SP)                                 ; 0835
                                 7E  D4 0002E               CLRL    -(SP)                                 ;
```

```
                               04  AC  DD 00030      PUSHL   LBN                                 ;
7E            10  AE               09  78 00033      ASHL    #9, FOUND_COUNT, -(SP)              ;
50                53               09  78 00038      ASHL    #9, I, R0                           ;
                              00 B440  9F 0003C      PUSHAB  @BUFFERS[R0]                        ;
                                   7E  7C 00040      CLRQ    -(SP)                               ;
                                   55  DD 00042      PUSHL   R5                                  ;
                                   21  DD 00044      PUSHL   #33                                 ;
                            0000G  CF  DD 00046      PUSHL   IO_CHANNEL                          ;
                                   01  DD 0004A      PUSHL   #1                                  ;
       00000000G  00               0C  FB 0004C      CALLS   #12, SYS$QIOW                       ;
                  03               50  E8 00053      BLBS    STATUS, 1$                          ; 0836
                  65               50  D0 00056      MOVL    STATUS, IO_STATUS                   ;
                  25               65  E8 00059 1$:  BLBS    IO_STATUS, 4$                       ; 0837
                  52               01  CE 0005C      MNEGL   #1, J                               ; 0840
                                   11  11 0005F      BRB     3$                                  ;
50                53               52  C1 00061 2$:  ADDL3   J, I, R0                            ; 0842
50                50               09  78 00065      ASHL    #9, R0, R0                          ;
                              00 B440  9F 00069      PUSHAB  @BUFFERS[R0]                        ;
           0000V  CF               01  FB 0006D      CALLS   #1, INVALIDATE                      ;
EB                52               6E  F2 00072 3$:  AOBLSS  FOUND_COUNT, J, 2$                  ;
                            0000G  CF  D4 00076      CLRL    DIR_VBN                             ; 0843
                            0000G  CF  D4 0007A      CLRL    BITMAP_VBN                          ; 0844
                                   65  BF 0007E      CHMU    IO_STATUS                           ; 0845
                                       04 00080      RET                                         ;
                  51               01  CE 00081 4$:  MNEGL   #1, J                               ; 0849
                                   0B  11 00084      BRB     6$                                  ;
50                53               51  C1 00086 5$:  ADDL3   J, I, R0                            ;
             F8 B440        0000G  CF  D0 0008A      MOVL    CURRENT_UCB, @BUFFER_UCB[R0]        ;
F1                51               6E  F2 00091 6$:  AOBLSS  FOUND_COUNT, J, 5$                  ;
50                53               09  78 00095 7$:  ASHL    #9, I, R0                           ; 0852
                  50               64  C0 00099      ADDL2   BUFFERS, R0                         ;
                                       04 0009C      RET                                         ; 0854

; Routine Size:  157 bytes    Routine Base:  $CODE$ + 02A9
;  546          0855  1 GLOBAL ROUTINE RESET_LBN (BUFFER, LBN) : NOVALUE =
;  547          0856  1
;  548          0857  1 !++
;  549          0858  1 !
;  550          0859  1 ! FUNCTIONAL DESCRIPTION:
;  551          0860  1 !
;  552          0861  1 !       This routine changes the resident LBN of the indicated block.
;  553          0862  1 !
;  554          0863  1 ! CALLING SEQUENCE:
;  555          0864  1 !       RESET_LBN (ARG1, ARG2)
;  556          0865  1 !
;  557          0866  1 ! INPUT PARAMETERS:
;  558          0867  1 !       ARG1: address of block buffer
;  559          0868  1 !       ARG2: new LBN
;  560          0869  1 !
;  561          0870  1 ! IMPLICIT INPUTS:
;  562          0871  1 !       buffer descriptor arrays
;  563          0872  1 !
;  564          0873  1 ! OUTPUT PARAMETERS:
;  565          0874  1 !       NONE
;  566          0875  1 !
;  567          0876  1 ! IMPLICIT OUTPUTS:
;  568          0877  1 !       NONE
;  569          0878  1 !
;  570          0879  1 ! ROUTINE VALUE:
;  571          0880  1 !       NONE
;  572          0881  1 !
;  573          0882  1 ! SIDE EFFECTS:
;  574          0883  1 !       backing LBN for buffer altered
;  575          0884  1 !
;  576          0885  1 !--
;  577          0886  1
;  578          0887  2 BEGIN
;  579          0888  2
;  580          0889  2 LOCAL
;  581          0890  2         I;                              ! index of buffer
;  582          0891  2
;  583          0892  2
;  584          0893  2 ! Compute the buffer index from the buffer address supplied. Set the
;  585          0894  2 ! buffer dirty bit and store the new LBN.
;  586          0895  2 !
;  587          0896  2
;  588          0897  2 IF .BUFFER LSSU BUFFERS[0] OR .BUFFER GEQU BUFFERS[.BUFFER_COUNT]
;  589          0898  2 THEN BUG_CHECK (BADBUFADR, FATAL, 'ACP buffer address out of range of buffer pool');
;  590          0899  2
;  591          0900  2 I = (.BUFFER - BUFFERS[0]) / 512;
;  592          0901  2 BUFFER_DIRTY[.I] = 1;
;  593          0902  2
;  594          0903  2 BUFFER_LBN[.I] = .LBN;
;  595          0904  2
;  596          0905  1 END;                                    ! end of routine RESET_LBN


                                                        .EXTRN  BUG$_BADBUFADR

                                        0004 00000      .ENTRY  RESET_LBN, Save R2                      ; 0855
```

```
                    52      0000'   CF  9E 00002          MOVAB   BUFFERS, R2                              ;
                    62         04   AC  D1 00007          CMPL    BUFFER, BUFFERS                          ; 0897
                                    0E  1F 0000B          BLSSU   1$                                       ;
    50       04     A2              09  78 0000D          ASHL    #9, BUFFER_COUNT, R0                     ;
                    50              62  C0 00012          ADDL2   BUFFERS, R0                              ;
                    50         04   AC  D1 00015          CMPL    BUFFER, R0                               ;
                                    04  1F 00019          BLSSU   2$                                       ;
                                      FEFF 0001B 1$:      BUGW                                             ; 0898
                                     0000* 0001D          .WORD   <BUG$_BADBUFADR!4>                       ;
    50       04     AC              62  C3 0001F 2$:      SUBL3   BUFFERS, BUFFER, R0                      ; 0900
                    50 00000200     8F  C6 00024          DIVL2   #512, I                                  ;
    00       FC     B2              50  E2 0002B          BBSS    I, @BUFFER_DIRTY, 3$                     ; 0901
               F4 B240         08   AC  D0 00030 3$:      MOVL    LBN, @BUFFER_LBN[I]                      ; 0903
                                        04 00036          RET                                              ; 0905

; Routine Size:  55 bytes,    Routine Base:  $CODE$ + 0346
```

```
1 GLOBAL ROUTINE WRITE_BLOCK (BUFFER) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine writes the indicated block back to the disk.
1 !
1 ! CALLING SEQUENCE:
1 !       WRITE_BLOCK (ARG1)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of block buffer
1 !
1 ! IMPLICIT INPUTS:
1 !       BUFFER DESCRIPTOR ARRAYS
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       block written
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       STATUS,                         ! service status of QIO call
2       I;                              ! index of buffer
2
2 EXTERNAL
2       PMS_TOT_WRITE,                  ! cumulative count of disk writes
2       CURRENT_UCB     : REF BBLOCK,   ! UCB of volume in process
2       DIR_VBN,                        ! current VBN in directory buffer
2       BITMAP_VBN,                     ! current VBN in storage map buffer
2       UNREC_COUNT,                    ! unrecorded but allocated blocks
2       NEW_FID,                        ! unrecorded new file ID
2       IO_CHANNEL,                     ! channel number for all I/O
2       IO_STATUS       : VECTOR,       ! status block for all I/O
2       CLEANUP_FLAGS   : BITVECTOR,    ! cleanup action flags
2       CONTEXT_SAVE    : BITVECTOR,    ! context save area
2       CONTEXT_START;                  ! start of reentrant context area
2
2
2 ! Compute the buffer index from the buffer address supplied. Clear the
2 ! buffer dirty bit and make sure the buffer ucb address corresponds to the
2 ! current UCB.
2 !
2
2 IF .BUFFER LSSU BUFFERS[0] OR .BUFFER GEQU BUFFERS[.BUFFER_COUNT]
```

```
;  655          0963  2 THEN BUG_CHECK (BADBUFADR, FATAL, 'ACP buffer address out of range of buffer pool');
;  656          0964  2
;  657          0965  2 I = (.BUFFER - BUFFERS[0]) / 512;
;  658          0966  2 BUFFER_DIRTY[.I] = 0;
;  659          0967  2
;  660          0968  2 IF .BUFFER_UCB[.I] NEQ .CURRENT_UCB
;  661          0969  2     THEN BUG_CHECK (WRTINVBUF, FATAL, 'ACP attempted to write an invalid buffer');
;  662          0970  2
;  663          0971  2 PMS_TOT_WRITE = .PMS_TOT_WRITE + 1;
;  664        P 0972  2 STATUS = $QIOW (
;  665        P 0973  2                 EFN  = EFN,
;  666        P 0974  2                 CHAN = .IO_CHANNEL,
;  667        P 0975  2                 FUNC = IO$_WRITELBLK,
;  668        P 0976  2                 IOSB = IO_STATUS,
;  669        P 0977  2                 P1   = BUFFERS[.I],
;  670        P 0978  2                 P2   = 512,
;  671        P 0979  2                 P3   = .BUFFER_LBN[.I]
;  672          0980  2                 );
;  673          0981  2
;  674          0982  2 ! If an I/O error occurrs, we must take special error handling. The first level
;  675          0983  2 ! handling currently implemented works for simple errors such as a write
;  676          0984  2 ! locked disk. It will not correctly unwind if successful writes have already
;  677          0985  2 ! occurred. We flush the cache of all buffers containing blocks from the current
;  678          0986  2 ! volume, and disable those portions of the cleanup that attempt to alter the
;  679          0987  2 ! disk.
;  680          0988  2 !
;  681          0989  2
;  682          0990  2 IF NOT .STATUS THEN IO_STATUS = .STATUS;
;  683          0991  2 IF NOT .IO_STATUS
;  684          0992  2 THEN
;  685          0993  3     BEGIN
;  686          0994  3     DIR_VBN = 0;
;  687          0995  3     BITMAP_VBN = 0;
;  688          0996  3     NEW_FID = 0;
;  689          0997  3     UNREC_COUNT = 0;
;  690          0998  3     CLEANUP_FLAGS = .CLEANUP_FLAGS AND NOT CLF_M_WRITEDISK;
;  691          0999  3     CLEANUP_FLAGS[CLF_FIXFCB] = 1;
;  692          1000  3     IF .CONTEXT_SAVE NEQ 0
;  693          1001  3     THEN
;  694          1002  4         BEGIN
;  695          1003  4         (CONTEXT_SAVE - CONTEXT_START + UNREC_COUNT) = 0;
;  696          1004  4         CONTEXT_SAVE = .CONTEXT_SAVE AND NOT CLF_M_WRITEDISK;
;  697          1005  4         CONTEXT_SAVE[CLF_FIXFCB] = 1;
;  698          1006  3         END;
;  699          1007  3     CH$FILL (0, (.BUFFER_COUNT+7)/8, BUFFER_DIRTY[0]);
;  700          1008  3     FLUSH_FID (0);
;  701          1009  3     ERR_EXIT (.IO_STATUS<0,16>);
;  702          1010  2     END;
;  703          1011  2
;  704          1012  1 END;                                    ! end of routine WRITE_BLOCK


                                                                         .EXTRN  PMS_TOT_WRITE, UNREC_COUNT
                                                                         .EXTRN  NEW_FID, CONTEXT_SAVE
                                                                         .EXTRN  CONTEXT_START, BUG$_WRTINVBUF
```

```
                                                         01FC 00000       .ENTRY  WRITE_BLOCK, Save R2,R3,R4,R5,R6,R7,R8        ; 0906
                                  58     0000G       CF    9E 00002       MOVAB   CONTEXT_SAVE, R8                              ;
                                  57     0000G       CF    9E 00007       MOVAB   IO_STATUS, R7                                 ;
                                  56     0000'       CF    9E 0000C       MOVAB   BUFFERS, R6                                   ;
                                  66        04       AC    D1 00011       CMPL    BUFFER, BUFFERS                               ; 0962
                                                     0E    1F 00015       BLSSU   1$                                            ;
                   50          04 A6                 09    78 00017       ASHL    #9, BUFFER_COUNT, R0                          ;
                                  50                 66    C0 0001C       ADDL2   BUFFERS, R0                                   ;
                                  50        04       AC    D1 0001F       CMPL    BUFFER, R0                                    ;
                                                     04    1F 00023       BLSSU   2$                                            ;
                                                         FEFF 00025 1$:   BUGW                                                  ; 0963
                                                        0000* 00027       .WORD   <BUG$_BADBUFADR!4>                            ;
                   50          04 AC                 66    C3 00029 2$:   SUBL3   BUFFERS, BUFFER, R0                           ; 0965
                                  50  00000200       8F    C6 0002E       DIVL2   #512, I                                       ;
                   00          FC B6                 50    E5 00035       BBCC    I, @BUFFER_DIRTY, 3$                          ; 0966
                            0000G CF            F8 B640    D1 0003A 3$:   CMPL    @BUFFER_UCB[I], CURRENT_UCB                   ; 0968
                                                     04    13 00041       BEQL    4$                                            ;
                                                         FEFF 00043       BUGW                                                  ; 0969
                                                        0000* 00045       .WORD   <BUG$_WRTINVBUF!4>                            ;
                                         0000G       CF    D6 00047 4$:   INCL    PMS_TOT_WRITE                                 ; 0971
                                                     7E    7C 0004B       CLRQ    -(SP)                                         ; 0980
                                                     7E    D4 0004D       CLRL    -(SP)                                         ;
                                                F4 B640    DD 0004F       PUSHL   @BUFFER_LBN[I]                                ;
                                  7E      0200       8F    3C 00053       MOVZWL  #512, -(SP)                                   ;
                   50             50                 09    78 00058       ASHL    #9, R0, R0                                    ;
                                                00 B640    9F 0005C       PUSHAB  @BUFFERS[R0]                                  ;
                                                     7E    7C 00060       CLRQ    -(SP)                                         ;
                                                     57    DD 00062       PUSHL   R7                                            ;
                                                     20    DD 00064       PUSHL   #32                                           ;
                                         0000G       CF    DD 00066       PUSHL   IO_CHANNEL                                    ;
                                                     01    DD 0006A       PUSHL   #1                                            ;
                        00000000G 00                 0C    FB 0006C       CALLS   #12, SYS$QIOW                                 ;
                                  03                 50    E8 00073       BLBS    STATUS, 5$                                    ; 0990
                                  67                 50    D0 00076       MOVL    STATUS, IO_STATUS                             ;
                                  4A                 67    E8 00079 5$:   BLBS    IO_STATUS, 7$                                 ; 0991
                                         0000G       CF    D4 0007C       CLRL    DIR_VBN                                       ; 0994
                                         0000G       CF    D4 00080       CLRL    BITMAP_VBN                                    ; 0995
                                         0000G       CF    D4 00084       CLRL    NEW_FID                                       ; 0996
                                         0000G       CF    D4 00088       CLRL    UNREC_COUNT                                   ; 0997
                            0000G CF  10FC0020       8F    CA 0008C       BICL2   #284950560, CLEANUP_FLAGS                     ; 0998
                            0000G CF                 02    88 00095       BISB2   #2, CLEANUP_FLAGS                             ; 0999
                                                     68    D5 0009A       TSTL    CONTEXT_SAVE                                  ; 1000
                                                     10    13 0009C       BEQL    6$                                            ;
                                     00000000*       EF    D4 0009E       CLRL    <<CONTEXT_SAVE-CONTEXT_START>+UNREC_COUNT>    ; 1003
                                  68  10FC0020       8F    CA 000A4       BICL2   #284950560, CONTEXT_SAVE                      ; 1004
                                  68                 02    88 000AB       BISB2   #2, CONTEXT_SAVE                              ; 1005
                   50          04 A6                 07    C1 000AE 6$:   ADDL3   #7, BUFFER_COUNT, R0                          ; 1007
                                  50                 08    C6 000B3       DIVL2   #8, R0                                        ;
           50      00             6E                 00    2C 000B6       MOVC5   #0, (SP), #0, R0, @BUFFER_DIRTY               ;
                                            FC       B6       000BB                                                         ;
                                                     7E    D4 000BD       CLRL    -(SP)                                         ; 1008
                            0000V CF                 01    FB 000BF       CALLS   #1, FLUSH_FID                                 ;
                                                     67    BF 000C4       CHMU    IO_STATUS                                     ; 1009
                                                           04 000C6 7$:   RET                                                   ; 1012

; Routine Size:  199 bytes,    Routine Base:  $CODE$ + 037D
;  705        1013  1
;  706        1014  1
;  707        1015  1 !++
;  708        1016  1 !
;  709        1017  1 !       The routine DIRPUT is equivalent to WRITE_BLOCK
;  710        1018  1 !
;  711        1019  1 !--
;  712        1020  1
;  713        1021  1 GLOBAL BIND ROUTINE
;  714        1022  1         DIRPUT          = WRITE_BLOCK;  ! write a directory record
```

```
1 GLOBAL ROUTINE CREATE_BLOCK (LBN, COUNT, TYPE, COUNT_FOUND) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine fabricates block buffer(s) containing the designated
1 !       block(s). The type code is as for READ_BLOCK and determines the buffer
1 !       pool to be used.
1 !
1 ! CALLING SEQUENCE:
1 !       CREATE_BLOCK (ARG1, ARG2, ARG3, ARG4)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: LBN to be assigned to block
1 !       ARG2: number of blocks to reserve in buffer
1 !       ARG3: block type code
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_UCB: UCB address of device in process
1 !
1 ! OUTPUT PARAMETERS:
1 !       ARG4: number of buffers found (optional)
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       address of buffer
1 !
1 ! SIDE EFFECTS:
1 !       buffer zeroed and recorded as a block read
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       I,                              ! index of buffer to use
2       FOUND_COUNT;                    ! number of buffers gotten
2
2 EXTERNAL
2       CURRENT_UCB     : REF BBLOCK;   ! address of device UCB
2
2
2 ! Find an available buffer. Mark it resident and dirty and fill it with
2 ! zeroes.
2 !
2
2 I = FIND_BUFFER (.LBN, .TYPE, .COUNT, FOUND_COUNT);
2 INCR J FROM 0 TO .FOUND_COUNT - 1
2 DO
3     BEGIN
3     BUFFER_UCB[.I+.J] = .CURRENT_UCB;
3     CH$FILL (0, 512, BUFFERS[.I+.J]);
3     BUFFER_DIRTY[.I+.J] = 1;
2     END;
```

```
;    773          1080  2
;    774          1081  2 IF ACTUALCOUNT GEQU 4
;    775          1082  2 THEN .COUNT_FOUND = .FOUND_COUNT;
;    776          1083  2 RETURN BUFFERS[.I];
;    777          1084  2
;    778          1085  1 END;                                      ! end of routine CREATE_BLOCK


                                                                     DIRPUT==              WRITE_BLOCK

                                                  01FC 00000              .ENTRY   CREATE_BLOCK, Save R2,R3,R4,R5,R6,R7,R8        ; 1023
                        5E                     04   C2 00002              SUBL2    #4, SP                                         ;
                                               5E   DD 00005              PUSHL    SP                                             ; 1072
                                08   AC             DD 00007              PUSHL    COUNT                                          ;
                                0C   AC             DD 0000A              PUSHL    TYPE                                           ;
                                04   AC             DD 0000D              PUSHL    LBN                                            ;
                  FCDD    CF                   04   FB 00010              CALLS    #4, FIND_BUFFER                                ;
                          58                   50   D0 00015              MOVL     R0, I                                          ;
                          57                   01   CE 00018              MNEGL    #1, J                                          ; 1078
                                               21   11 0001B              BRB      2$                                             ;
              56          58                   57   C1 0001D 1$:          ADDL3    J, I, R6                                       ; 1076
           0000'DF46             0000G    CF        D0 00021              MOVL     CURRENT_UCB, @BUFFER_UCB[R6]                   ;
              50          56                   09   78 00029              ASHL     #9, R6, R0                                     ; 1077
 0200    8F   00          6E                   00   2C 0002D              MOVC5    #0, (SP), #0, #512, @BUFFERS[R0]               ;
                                 0000'DF40             00034                                                                     ;
              00   0000'  DF                   56   E2 00038              BBSS     R6, @BUFFER_DIRTY, 2$                          ; 1078
              DB          57                   6E   F2 0003E 2$:          AOBLSS   FOUND_COUNT, J, 1$                             ; 1073
                          04                   6C   91 00042              CMPB     (AP), #4                                       ; 1081
                                               04   1F 00045              BLSSU    3$                                             ;
                     10   BC                   6E   D0 00047              MOVL     FOUND_COUNT, @COUNT_FOUND                      ; 1082
              50          58                   09   78 0004B 3$:          ASHL     #9, I, R0                                      ; 1083
                          50     0000'    CF        C0 0004F              ADDL2    BUFFERS, R0                                    ;
                                                    04 00054              RET                                                     ; 1085

; Routine Size:  85 bytes,    Routine Base:  $CODE$ + 0444
;  780        1086  1 GLOBAL ROUTINE MARK_DIRTY (BUFFER) : NOVALUE =
;  781        1087  1
;  782        1088  1 !++
;  783        1089  1 !
;  784        1090  1 ! FUNCTIONAL DESCRIPTION:
;  785        1091  1 !
;  786        1092  1 !       This routine marks the indicated buffer for write-back.
;  787        1093  1 !
;  788        1094  1 ! CALLING SEQUENCE:
;  789        1095  1 !       MARK_DIRTY (ARG1)
;  790        1096  1 !
;  791        1097  1 ! INPUT PARAMETERS:
;  792        1098  1 !       ARG1: address of block buffer
;  793        1099  1 !
;  794        1100  1 ! IMPLICIT INPUTS:
;  795        1101  1 !       NONE
;  796        1102  1 !
;  797        1103  1 ! OUTPUT PARAMETERS:
;  798        1104  1 !       NONE
;  799        1105  1 !
;  800        1106  1 ! IMPLICIT OUTPUTS:
;  801        1107  1 !       NONE
;  802        1108  1 !
;  803        1109  1 ! ROUTINE VALUE:
;  804        1110  1 !       NONE
;  805        1111  1 !
;  806        1112  1 ! SIDE EFFECTS:
;  807        1113  1 !       buffer marked for write-back
;  808        1114  1 !
;  809        1115  1 !--
;  810        1116  1
;  811        1117  2 BEGIN
;  812        1118  2
;  813        1119  2 LOCAL
;  814        1120  2       I;                              ! index of buffer
;  815        1121  2
;  816        1122  2
;  817        1123  2 IF .BUFFER LSSU BUFFERS[0] OR .BUFFER GEQU BUFFERS[.BUFFER_COUNT]
;  818        1124  2 THEN BUG_CHECK (BADBUFADR, FATAL, 'ACP buffer address out of range of buffer pool');
;  819        1125  2
;  820        1126  2 I = (.BUFFER - BUFFERS[0]) / 512;
;  821        1127  2
;  822        1128  2 BUFFER_DIRTY[.I] = 1;
;  823        1129  2
;  824        1130  1 END;                                  ! end of routine MARK_DIRTY


                                  0004 00000        .ENTRY  MARK_DIRTY, Save R2                       ; 1086
                  52      0000'  CF  9E 00002        MOVAB   BUFFERS, R2                               ; 1123
                  62         04  AC  D1 00007        CMPL    BUFFER, BUFFERS                           ;
                                 0E  1F 0000B        BLSSU   1$                                        ;
          50      04   A2            09  78 0000D     ASHL    #9, BUFFER_COUNT, R0                      ;
                  50                 62  C0 00012     ADDL2   BUFFERS, R0                               ;
                  50         04  AC  D1 00015        CMPL    BUFFER, R0                                ;
```

```
                                                04  1F 00019        BLSSU   2$                                         ;
                                                 FEFF 0001B 1$:     BUGW                                               ; 1124
                                                0000* 0001D         .WORD   <BUG$_BADBUFADR!4>                         ;
                        50        04    AC      62  C3 0001F 2$:     SUBL3   BUFFERS, BUFFER, R0                        ; 1126
                                        50 00000200  8F  C6 00024   DIVL2   #512, I                                    ;
                        00        FC    B2      50  E2 0002B         BBSS    I, @BUFFER_DIRTY, 3$                       ; 1128
                                                    04 00030 3$:     RET                                                ; 1130

; Routine Size:  49 bytes,    Routine Base:  $CODE$ + 0499
```

```
1 GLOBAL ROUTINE INVALIDATE (BUFFER) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine invalidates the indicated buffer.
1 !
1 ! CALLING SEQUENCE:
1 !       INVALIDATE (ARG1)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of block buffer
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       buffer contents forgotten
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       I,                                      ! index of buffer
2       POOL,                                   ! index of pool
2       LRU_ENTRY;                              ! address of LRU list entry
2
2 ! A buffer is invalidated by zeroing its associated UCB address and
2 ! clearing the dirty bit. Also, we relink the buffer onto the front of the
2 ! buffer LRU to encourage its re-use.
2 !
2
2 IF .BUFFER LSSU BUFFERS[0] OR .BUFFER GEQU BUFFERS[.BUFFER_COUNT]
2 THEN BUG_CHECK (BADBUFADR, FATAL, 'ACP buffer address out of range of buffer pool');
2
2 I = (.BUFFER - BUFFERS[0]) / 512;
3 POOL = (
3     INCR J FROM 0 TO POOL_COUNT-1 DO
3     IF .I LSS .POOL_BASE[.J] + .POOL_SIZE[.J]
3     THEN EXITLOOP .J
2     );
2
2 BUFFER_UCB[.I] = 0;
2 BUFFER_DIRTY[.I] = 0;
2
```

```
;  883          1188  2 REMQUE (BUFFER_LRU[.I, LRU_FLINK], LRU_ENTRY);
;  884          1189  2 INSQUE (.LRU_ENTRY, POOL_LRU[.POOL, LRU_FLINK]);
;  885          1190  2
;  886          1191  1 END;                                      ! end of routine INVALIDATE



                                       001C 00000         .ENTRY   INVALIDATE, Save R2,R3,R4                          ; 1131
                    54      0000'   CF   9E 00002         MOVAB    BUFFERS, R4                                        ;
                    64         04   AC   D1 00007         CMPL     BUFFER, BUFFERS                                    ; 1175
                                    0E   1F 0000B         BLSSU    1$                                                 ;
          50      04   A4           09   78 0000D         ASHL     #9, BUFFER_COUNT, R0                               ;
                    50              64   C0 00012         ADDL2    BUFFERS, R0                                        ;
                    50         04   AC   D1 00015         CMPL     BUFFER, R0                                         ;
                                    04   1F 00019         BLSSU    2$                                                 ;
                                       FEFF 0001B 1$:     BUGW                                                        ; 1176
                                      0000* 0001D         .WORD    <BUG$_BADBUFADR!4>                                 ;
          50      04   AC           64   C3 0001F 2$:     SUBL3    BUFFERS, BUFFER, R0                                ; 1178
          51           50 00000200  8F   C7 00024         DIVL3    #512, R0, I                                        ;
                                    50   D4 0002C         CLRL     J                                                  ; 1181
                    52       C4 A440     3C 0002E 3$:     MOVZWL   POOL_BASE[J], R2                                   ;
                    53       CC A440     3C 00033         MOVZWL   POOL_SIZE[J], R3                                   ;
                    52              53   C0 00038         ADDL2    R3, R2                                             ;
                    52              51   D1 0003B         CMPL     I, R2                                              ;
                                    07   19 0003E         BLSS     4$                                                 ;
          EA           50           02   F3 00040         AOBLEQ   #2, J, 3$                                          ;
                       50           01   CE 00044         MNEGL    #1, POOL                                           ; 1180
                             F8 B441     D4 00047 4$:     CLRL     @BUFFER_UCB[I]                                     ; 1185
          00      FC   B4           51   E5 0004B         BBCC     I, @BUFFER_DIRTY, 5$                               ; 1186
                    51       EC B441     7E 00050 5$:     MOVAQ    @BUFFER_LRU[I], R1                                 ; 1188
                    52              61   0F 00055         REMQUE   (R1), LRU_ENTRY                                    ;
                    50       D4 A440     7E 00058         MOVAQ    POOL_LRU[POOL], R0                                 ; 1189
                    60              62   0E 0005D         INSQUE   (LRU_ENTRY), (R0)                                  ;
                                         04 00060         RET                                                         ; 1191

; Routine Size:  97 bytes,    Routine Base:  $CODE$ + 04CA
```

```
1 GLOBAL ROUTINE WRITE_HEADER : NOVALUE  =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine writes out the currently resident file header.
1 !
1 ! CALLING SEQUENCE:
1 !       WRITE_HEADER ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       FILE_HEADER: address of current file header
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       IO_STATUS: status of I/O transfer
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       checksum checked, header written
1 !
1 !--
1
2 BEGIN
2
2 EXTERNAL
2       FILE_HEADER     : REF BBLOCK;   ! address of last file header read
2
2 EXTERNAL ROUTINE
2       CHECKSUM;                       ! compute file header checksum
2
2
2 ! The checksum of the header should be good, since all routines that modify
2 ! the header bless it with a new checksum when they are finished. Check the
2 ! checksum and write the header.
2 !
2
2 IF NOT CHECKSUM (.FILE_HEADER)
2 THEN BUG_CHECK (WRTINVHDR, FATAL, 'ACP attempted to write an invalid file header');
2
2 WRITE_BLOCK (.FILE_HEADER);
2
1 END;                                  ! end of routine WRITE_HEADER


                                                        .EXTRN  FILE_HEADER, CHECKSUM
                                                        .EXTRN  BUG$_WRTINVHDR
```

```
                                          0000 00000          .ENTRY  WRITE_HEADER, Save nothing                              ; 1192
                             0000G   CF   DD 00002            PUSHL   FILE_HEADER                                             ; 1237
                0000G   CF           01   FB 00006            CALLS   #1, CHECKSUM                                            ;
                   04                50   E8 0000B            BLBS    R0, 1$                                                  ;
                                          FEFF 0000E          BUGW                                                            ; 1238
                                          0000* 00010         .WORD   <BUG$_WRTINVHDR!4>                                      ;
                             0000G   CF   DD 00012 1$:        PUSHL   FILE_HEADER                                             ; 1240
                FE37    CF           01   FB 00016            CALLS   #1, WRITE_BLOCK                                         ;
                                          04 0001B            RET                                                             ; 1242

; Routine Size:  28 bytes,    Routine Base:  $CODE$ + 052B
;  940          1243  1 GLOBAL ROUTINE FLUSH_BUFFERS : NOVALUE =
;  941          1244  1
;  942          1245  1 !++
;  943          1246  1 !
;  944          1247  1 ! FUNCTIONAL DESCRIPTION:
;  945          1248  1 !
;  946          1249  1 !       This routine writes all buffers which were modified back to the
;  947          1250  1 !       disk from whence they came.
;  948          1251  1 !
;  949          1252  1 ! CALLING SEQUENCE:
;  950          1253  1 !       FLUSH_BUFFERS[0] ()
;  951          1254  1 !
;  952          1255  1 ! INPUT PARAMETERS:
;  953          1256  1 !       NONE
;  954          1257  1 !
;  955          1258  1 ! IMPLICIT INPUTS:
;  956          1259  1 !       all own storage of this module
;  957          1260  1 !
;  958          1261  1 ! OUTPUT PARAMETERS:
;  959          1262  1 !       NONE
;  960          1263  1 !
;  961          1264  1 ! IMPLICIT OUTPUTS:
;  962          1265  1 !       NONE
;  963          1266  1 !
;  964          1267  1 ! ROUTINE VALUE:
;  965          1268  1 !       NONE
;  966          1269  1 !
;  967          1270  1 ! SIDE EFFECTS:
;  968          1271  1 !       dirty buffers written.
;  969          1272  1 !
;  970          1273  1 !--
;  971          1274  1
;  972          1275  2 BEGIN
;  973          1276  2
;  974          1277  2
;  975          1278  2 ! We simply scan the dirty bit vector and write all buffers marked dirty.
;  976          1279  2 !
;  977          1280  2
;  978          1281  2 INCR I FROM 0 TO .BUFFER_COUNT-1 DO
;  979          1282  2     IF .BUFFER_DIRTY[.I]
;  980          1283  2     THEN WRITE_BLOCK (BUFFERS[.I]);
;  981          1284  2
;  982          1285  1 END;                                   ! end of routine FLUSH_BUFFERS[0]



                                 000C 00000           .ENTRY  FLUSH_BUFFERS, Save R2,R3                  ; 1243
              53     0000'   CF    D0 00002           MOVL    BUFFER_COUNT, R3                           ; 1281
              52             01    CE 00007           MNEGL   #1, I                                      ;
                             14    11 0000A           BRB     2$                                         ;
 0E     0000' DF             52    E1 0000C 1$:       BBC     I, @BUFFER_DIRTY, 2$                       ; 1282
 50           52             09    78 00012           ASHL    #9, I, R0                                  ; 1283
                     0000'DF40     9F 00016           PUSHAB  @BUFFERS[R0]                               ;
        FE16  CF             01    FB 0001B           CALLS   #1, WRITE_BLOCK                            ;
 E8           52             53    F2 00020 2$:       AOBLSS  R3, I, 1$                                  ; 1282
```

```
                                                       04 00024          RET                                                  ; 1285

; Routine Size:  37 bytes,     Routine Base:  $CODE$ + 0547
```

```
1 GLOBAL ROUTINE FLUSH_FID (FID) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine removes from the buffer cache all blocks contained
1 !       within the specified file. Dirty buffers are written.
1 !
1 ! CALLING SEQUENCE:
1 !       FLUSH_FID (ARG1)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: file ID of file to flush
1 !             0 to match all
1 !
1 ! IMPLICIT INPUTS:
1 !       all own storage of this module
1 !       CURRENT_UCB: UCB of current device
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       NONE
1 !
1 ! SIDE EFFECTS:
1 !       dirty buffers written, appropriate buffers invalidated
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       FID             : REF BBLOCK;   ! file ID arg
2 LOCAL
2       I;                              ! index to buffers
2
2 EXTERNAL
2       CURRENT_UCB     : REF BBLOCK,   ! address of device UCB
2       CURRENT_VCB     : REF BBLOCK;   ! address of current VCB
2
2
2 ! We scan the UCB and FID vectors looking for matches. Buffers that match
2 ! are written if dirty and then invalidated.
2 !
2
2 INCR I FROM 0 TO .BUFFER_COUNT-1 DO
3     BEGIN
3     IF .BUFFER_UCB[.I] EQL .CURRENT_UCB
4     AND (.FID EQL 0
5         OR  (.(BUFFER_FID[.I])<0,16> EQL .FID[FID$W_NUM]
6             AND (IF .CURRENT_VCB[VCB$V_EXTFID]
6                 THEN .(BUFFER_FID[.I])<16,8> EQL .FID[FID$B_NMX]
; 1041        1343 6                  ELSE 1
; 1042        1344 6                  )
; 1043        1345 5              )
; 1044        1346 4          )
; 1045        1347 3      THEN
; 1046        1348 4          BEGIN
; 1047        1349 4          IF .BUFFER_DIRTY[.I]
; 1048        1350 4          THEN WRITE_BLOCK (BUFFERS[.I]);
; 1049        1351 4          INVALIDATE (BUFFERS[.I]);
; 1050        1352 3          END;
; 1051        1353 2      END;
; 1052        1354 2
; 1053        1355 1 END;                                    ! end of routine FLUSH_FID



                                                001C 00000         .ENTRY   FLUSH_FID, Save R2,R3,R4                          ; 1286
                        54      0000'   CF      9E 00002         MOVAB    BUFFER_FID, R4                                    ;
                        53        14    A4      D0 00007         MOVL     BUFFER_COUNT, R3                                  ; 1336
                        52              01      CE 0000B         MNEGL    #1, I                                             ;
                                        50      11 0000E         BRB      4$                                                ;
               0000G    CF      08 B442         D1 00010 1$:     CMPL     @BUFFER_UCB[I], CURRENT_UCB                       ; 1338
                                        47      12 00017         BNEQ     4$                                                ;
                        51        04    AC      D0 00019         MOVL     FID, R1                                           ; 1339
                                        22      13 0001D         BEQL     2$                                                ;
                                00 B442         DF 0001F         PUSHAL   @BUFFER_FID[I]                                    ; 1340
                        61              9E      B1 00023         CMPW     @(SP)+, (R1)                                      ;
                                        38      12 00026         BNEQ     4$                                                ;
                        50      0000G   CF      D0 00028         MOVL     CURRENT_VCB, R0                                   ; 1341
           0F       0B  A0              05      E1 0002D         BBC      #5, 11(R0), 2$                                    ;
                        50        05    A1      9A 00032         MOVZBL   5(R1), R0                                         ; 1342
                                00 B442         DF 00036         PUSHAL   @BUFFER_FID[I]                                    ;
   50      9E               08          10      ED 0003A         CMPZV    #16, #8, @(SP)+, R0                               ;
                                        1F      12 0003F         BNEQ     4$                                                ;
           0D       0C  B4              52      E1 00041 2$:     BBC      I, @BUFFER_DIRTY, 3$                              ; 1349
           50           52              09      78 00046         ASHL     #9, I, R0                                         ; 1350
                                10 B440         9F 0004A         PUSHAB   @BUFFERS[R0]                                      ;
               FDBE     CF              01      FB 0004E         CALLS    #1, WRITE_BLOCK                                   ;
           50           52              09      78 00053 3$:     ASHL     #9, I, R0                                         ; 1351
                                10 B440         9F 00057         PUSHAB   @BUFFERS[R0]                                      ;
               FEFE     CF              01      FB 0005B         CALLS    #1, INVALIDATE                                    ;
           AC           52              53      F2 00060 4$:     AOBLSS   R3, I, 1$                                         ; 1336
                                                04 00064         RET                                                        ; 1355

; Routine Size:  101 bytes,    Routine Base:  $CODE$ + 056C


; 1054        1356 1
; 1055        1357 1 END
; 1056        1358 0 ELUDOM
;                                       PSECT SUMMARY
;
;       Name                        Bytes                        Attributes
;
;   $CODE$                             1489  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $LOCKEDD1$                           68  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)




;                                 Library Statistics
;
;                                              -------- Symbols --------        Pages        Processing
;       File                                   Total    Loaded  Percent         Mapped       Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1              18619      16        0           1000           00:01.9




;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:RDBLOK/OBJ=OBJ$:RDBLOK MSRC$:RDBLOK/UPDATE=(ENH$:RDBLOK)

; Size:          1484 code + 73 data bytes
; Run Time:         00:28.7
; Elapsed Time:     01:07.3
; Lines/CPU Min:     2841
; Lexemes/CPU-Min: 16229
; Memory Used:  176 pages
; Compilation Complete
```

0166 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY